NAME

getprotobyname, getprotobyname_r, getprotobynumber, getprotobynumber_r, getprotoent, getprotoent_r, setprotoent, endprotoent – get protocol entry

SYNOPSIS

cc [*flag* ...] *file* ... **–lsocket -lnsl** [*library* ...]

#include <netdb.h>

struct protoent *getprotobyname(const char ***name*);**

struct protoent *getprotobyname_r(const char ***name,* **struct protoent ***result,* **char ***buffer,* **int *buflen***);**

struct protoent *getprotobynumber(int *proto***);**

struct protoent *getprotobynumber_r(int *proto,* **struct protoent ***result,* **char ***buffer,* **int** *buflen***);**

struct protoent *getprotoent(void);

struct protoent *getprotoent_r(struct protoent ***result,* **char ***buffer,* **int *buflen***);**

int setprotoent(int *stayopen***);**

int endprotoent(void);

MT-LEVEL

See the **NOTES** section of this page.

DESCRIPTION

These routines return a protocol entry. Two types of interfaces are supported: reentrant (**getprotobyname_r( )**, **getprotobynumber_r( )**, and **getprotoent_r( )**) and non-reentrant (**getprotobyname( )**, **getprotobynumber( )**, and **getprotoent( )**). The reentrant routines may be used in single-threaded applications and are safe for multi-threaded applications, making them the preferred interfaces.

The reentrant routines require additional parameters which are used to return results data. *result* is a pointer to a **struct protoent** structure and will be where the returned results will be stored. *buffer* is used as storage space for elements of the returned results. *buflen* is the size of *buffer* and should be large enough to contain all returned data. *buflen* must be at least 1024 bytes.

getprotobyname_r( ), **getprotobynumber_r( )**, and **getprotoent_r( )** each return a protocol entry.

The entry may come from one of the following sources: the protocols file (see **protocols**(4)), the NIS maps ''protocols.byname'' and ''protocols.bynumber'', and the NIS+ table ''protocols''. The sources and their lookup order are specified in the **/etc/nsswitch.conf** file (see **nsswitch.conf**(4) for details). Some name services such as NIS will return only one name for a host, whereas others such as NIS+ or DNS will return all aliases.

getprotobyname_r( ) and **getprotobynumber_r( )** sequentially search from the beginning of the file until a matching protocol name or protocol number is found, or until an EOF is encountered.

getprotobyname( ) and **getprotobynumber( )** have the same functionality as **getprotobyname_r( )** and **getprotobynumber_r( )** except that a static buffer is used to store returned results. These routines are unsafe in a multi-threaded application.

getprotoent_r( ) enumerates protocol entries: successive calls to **getprotoent_r( )** will return either successive protocol entries or NULL. Enumeration may not be supported by some sources. Note that if multiple threads call **getprotoent_r( )**, each will retrieve a subset of the protocol database.

getprotent( ) has the same functionality as **getprotent_r( )** except that a static buffer is used to store returned results. This routine is unsafe in a multi-threaded application.

setprotoent( ) "rewinds" to the beginning of the enumeration of protocol entries. If the *stayopen* flag is non-zero, resources such as open file descriptors are not deallocated after each call to **getprotobynumber_r( )** and **getprotobyname_r( )**. Calls to **getprotobyname_r( ) , getprotobyname( ) , getprotobynumber_r( )** and **getprotobynumber( )** may leave the enumeration in an indeterminate state, so **setprotoent( )** should be called before the first **getprotoent_r( )** or **getprotoent( )**. Note that **setprotoent( )** has process-wide scope, and ''rewinds'' the protocol entries for all threads calling **getprotoent_r( )** as well as main-thread calls to **getprotoent( )**.

endprotoent( ) may be called to indicate that protocol processing is complete; the system may then close any open protocols file, deallocate storage, and so forth. It is legitimate, but possibly less efficient, to call more protocol routines after **endprotoent( )**.

The internal representation of a protocol entry is a **protoent** structure defined in **<netdb.h>** with the following members:

```
char    *p_name;
char    **p_aliases;
int     p_proto;
```

RETURN VALUES

getprotobyname_r( ), getprotobyname( ), getprotobynumber_r( ), and **getprotobynumber( )** return a pointer to a **struct protoent** if they successfully locate the requested entry; otherwise they return NULL.

getprotoent_r( ) and **getprotoent( )** return a pointer to a **struct protoent** if they successfully enumerate an entry; otherwise they return NULL, indicating the end of the enumeration.

ERRORS

getprotobyname_r( ), getprotobynumber_r( ), and **getprotoent_r( )** will fail if the following is true:

ERANGE — length of the buffer supplied by caller is not large enough to store the result.

FILES

/etc/protocols
/etc/nsswitch.conf

SEE ALSO

intro(3), **nsswitch.conf**(4), **protocols**(4)

NOTES

Although **getprotobyname_r( )**, **getprotobynumber_r( )**, and **getprotoent_r( )** are not mentioned by POSIX.4a Draft 6, they were added to complete the functionality provided by similar thread-safe functions. These interfaces are subject to change to be compatible with the "spirit" of POSIX.4a when it is approved as a standard.

When compiling multithreaded applications, see **intro**(3), *Notes On Multithread Applications*, for information about the use of the **_REENTRANT** flag.

The routines **getprotobyname_r( )**, **getprotobynumber_r( )**, and **getprotoent_r( )** are reentrant and multi-thread safe. The reentrant interfaces can be used in single-threaded as well as multi-threaded applications and are therefore the preferred interfaces.

The routines **getprotobyname( )**, **getprotobyaddr( )**, and **getprotoent( )** use static storage, so returned data must be copied if it is to be saved. Because of their use of static storage for returned data, these routines are not safe for multi-threaded applications.

setprotoent( ) and **endprotoent( )** have process-wide scope, and are therefore not safe in multi-threaded applications.

Use of **getprotoent_r()** and **getprotoent()** is discouraged; enumeration is well-defined for the protocols file and is supported (albeit inefficiently) for NIS and NIS+, but in general may not be well-defined. The semantics of enumeration are discussed in **nsswitch.conf**(4).

BUGS

Only the Internet protocols are currently understood.

Programs that call **getprotobyname_r( )** or **getprotobynumber_r( )** routines cannot be linked statically since the implementation of these routines requires dynamic linker functionality to access shared objects at run time.